A General Reduction Theorem with Applications to Pathwidth and the Complexity of MAX 2-CSP

Edwards, Keith; McDermid, Eric

Published in:
Algorithmica

DOI:
10.1007/s00453-014-9883-7

Publication date:
2015

Document Version
Peer reviewed version

A General Reduction Theorem with Applications to Pathwidth and the Complexity of MAX 2-CSP

Edwards, Keith; McDermid, Eric

Published in:
Algorithmica

DOI:
10.1007/s00453-014-9883-7

Publication date:
2014

The final publication is available at Springer via http://dx.doi.org/10.1007/s00453-014-9883-7

A general reduction theorem with applications to pathwidth and the complexity of MAX 2-CSP

Keith Edwards
School of Computing
University of Dundee
Dundee, DD1 4HN
U.K.
kjedwards@dundee.ac.uk

Eric McDermid
Austin, TX
U.S.A.
em4617@gmail.com

April 29, 2014

Abstract

We prove a general reduction theorem which allows us to extend bounds for certain graph parameters on cubic graphs to bounds for general graphs taking into account the individual vertex degrees. As applications, we give an algorithm for Max 2-CSP whose complexity matches the algorithm of Scott and Sorkin in the case of d-regular graphs, $d \leq 5$, but is otherwise faster. It also improves on the previously fastest known algorithm in terms of the average degree, given by Golovnev and Kutzkov. Also from the general theorem, we derive a bound for the pathwidth of a general graph which equals that of Fomin *et al.* and Gaspers for graphs of degree at most 6, but is smaller otherwise, and use this to give an improved exponential-space algorithm for Max 2-CSP. Finally we use the general result to give a faster algorithm for Max 2-CSP on claw-free graphs.

Keywords: constraint satisfaction problems; Max 2-CSP; treewidth; pathwidth.

1 Introduction

For over 30 years, there has been interest in the design of faster exponential-time algorithms for NP-hard problems. On the one hand, there are a number of problems for which significant progress has been made. A good example of this is maximum independent set – the best[1] algorithm solves this problem in $O^\star(1.2002^n)$ time and polynomial space [33]. Many other classical NP-hard combinatorial problems can also be solved much faster than $O^\star(2^n)$; we refer the reader to the excellent surveys of Woeginger [31, 32] or the book by Fomin and Kratsch [13] for further examples.

On the other hand, there are a number of cut, colouring, and partition problems, such as Max Cut, Max 2-Sat, and Max Bisection, for example, that have proven to be tougher nuts to crack. Currently there is no known polynomial-space algorithm with time complexity better than $O^\star(2^n)$ for any of these problems. Hence, a recent line of research has been to instead design algorithms for *sparse* instances of such problems. In this setting, the algorithms are parameterized by m, the number of edges (or, in the case of Max 2-Sat, the number of clauses). The time complexities of such algorithms take the form $O^\star(2^{m/c})$, where c is a constant. Of course, such an algorithm makes an implicit assumption that the average degree of the graph (or average number of appearances of a variable) is bounded by $2c$ – otherwise the naive $O^\star(2^n)$-time algorithm is faster.

A standard technique in designing moderately exponential-time algorithms is the *branch-and-reduce* paradigm. A branch-and-reduce algorithm uses a set of *reduction rules*, which repeatedly transform the instance into a simpler, equivalent instance. Next, when no reduction rule applies, the algorithm uses a set of *branching rules* to generate a collection of instances which are then solved recursively. If one looks back over the history of exponential time algorithms designed for, say, Max 2-Sat or Max 2-CSP, the successive improvements in time complexity usually come at the cost of more and more sophisticated reduction rules, which often also require more and more sophisticated analysis techniques. Similar reductions are used by various authors [10, 14, 21, 28] to obtain bounds on the treewidth and pathwidth of a general graph.

1.1 Known results

There has been a long sequence of exponential time algorithms parameterized by m for problems contained in Max 2-CSP, particularly for the special case Max 2-Sat. Currently the best polynomial-space algorithm for Max 2-CSP parameterized by m is due to Scott and Sorkin [28] and Gaspers and Sorkin [15], who describe an algorithm ("Algorithm B") with time complexity $O^\star(r^{19m/100}) = O^\star(r^{m/5.623})$ (where r is the domain size for each variable). (Algorithm A was a simpler algorithm which we do not consider here.)

There have also been algorithms parameterized by the average degree d and number of vertices n. Scott and Sorkin [28] give an algorithm which for graphs of average degree d requires time at most $O^\star(r^{(1-\frac{2}{d+1})n})$. This is improved to $O^\star(r^{(1-\frac{3}{d+1})n})$ by Golovnev and Kutzkov [17] (we refer to this as Algorithm D). An earlier version of this paper [16]

[1] There is also a technical report, due to Robson [26], which makes use of a detailed analysis done by computer to solve maximum independent set in $O^\star(2^{.25n})$ time and exponential space.

claimed bounds similar to ours, but these are not included in the journal paper [17]. Golovnev and Kutzkov [17] also give an algorithm which for very large average degree d has a smaller exponent of the form $(1 - O(\frac{\ln d}{d}))n$.

For treewidth, Scott and Sorkin [28] derive a general upper bound of $(13/75 + o(1))m$ for a graph with m edges, and Kneis *et al.* use similar methods to obtain an upper bound of $(13/75 + o(1))m + O(\log n)$ on the pathwidth of a graph. Fomin *et al.* [10] show that

$$\mathrm{pw}(G) \leq \tfrac{1}{6}n_3 + \tfrac{1}{3}n_4 + \tfrac{13}{30}n_5 + \tfrac{23}{45}n_6 + n_{\geq 7} + o(n)$$

where n_i is the number of vertices of degree i, and $n_{\geq 7}$ is the number of vertices of degree at least 7. A similar bound with the sum extending to degree 17 is given by Gaspers [14], though in this case the coefficients of the numbers n_i are found by computer search and are not given exactly.

Scott and Sorkin use their bound to give an algorithm (Algorithm C) for Max 2-CSP requiring time $O^\star(r^{(13/75+o(1))m})$ and exponential space.

For Max 2-Sat, the more specific nature of the problem allows additional types of reduction to be used, allowing somewhat faster algorithms. A series of papers have obtained running times of $O^\star(r^{m/c})$, with gradually increasing values of c, using increasingly sophisticated sets of reductions: $c = 2.879$ by Neidermeier and Rossmanith [24], $c = 3.448$ by Bansal and Raman [2], $c = 4$ by Hirsch [20], $c = 5$ by Gramm *et al.* [18], $c = 5.263$ by Scott and Sorkin [28], $c = 5.5$ by Kojevnikov and Kulikov [22], $c = 5.88$ by Kulikov and Kutzkov [23], $c = 6.215$ by Raible and Fernau [25], and finally $c = 6.321$ by Gaspers and Sorkin [15]. While the earlier algorithms are beaten by the current fastest algorithms for Max 2-CSP, the most recent algorithms for Max 2-Sat have taken advantage of the special nature of Max 2-Sat to achieve faster running times that, so far at least, have not been matched for general Max 2-CSP.

For the Max Cut problem, an $O^\star(2^{(1-2/d)n})$ algorithm is given by Della Croce, Kaminski and Paschos [6].

1.2 Our contribution

As mentioned above, a number of results have used increasingly complex reductions to obtain upper bounds. These reductions have mostly focussed on vertices of small degree (usually at most 5 or 6). Also, although this fact may often be obscured by the derivation of the reductions chosen, the best upper bounds have often used a vertex of maximum degree with, where possible, a neighbour of lower degree. This means that the case of regular graphs is a special case, which has been dealt with by various often rather complex means, or, in some cases, incorrectly.

In this paper we do three key things: (1) Rather than using reduction to prove a bound for a particular parameter, we prove a general reduction result using a function g_α which takes into account the degree of each vertex. Upper bounds for particular parameters can then be derived from this in a more or less standard way - we give three examples of this. We also fully analyse the behaviour of g_α. (2) We do not focus on small vertex degrees, but treat all degrees ≥ 4 the same, using a vertex of maximum degree with (where possible) a neighbour of lower degree. The introduction (as in [8]) of a small negative

term into the upper bound deals with the regular case and allows a fairly straightforward inductive proof. (3) Our algorithm for Max 2-CSP finds a set whose deletion from the graph leaves a graph of treewidth 3 rather than a series-parallel graph (treewidth 2). This allows the introduction of the small negative term in the cubic case, so that the general theorem can be applied. Our analysis of the function g_α gives the improved exponent of $(1 - \frac{13/4}{d+1} + O(1/d^3))n$.

We also derive a similar but better bound in the case of claw-free graphs, allowing a faster algorithm in this case.

Next we consider pathwidth, and use our general theorem to obtain an upper bound on the pathwidth of a graph in terms of the degrees of its vertices. This allows us to remove the error terms $n_{\geq 7}$ or $n_{\geq 18}$ in the upper bounds of Fomin *et al.* [10] and Gaspers [14] and give a sum over all $d \geq 3$, with the coefficients explicitly defined.

Scott and Sorkin use a bound on treewidth to give an algorithm for Max 2-CSP with lower exponential time complexity, but also requiring exponential space. Our improved bound thus gives faster bounds for the complexity of this algorithm.

Finally we consider whether reducing the constraint graph to a graph of larger, but still bounded, treewidth, can give still faster polynomial space algorithms for Max 2-CSP. Using the concept of fragmentability, we are able to give a bound on how much might be gained by this technique.

2 Preliminaries

All graphs $G = (V, E)$ are considered to be simple. We use standard graph terminology and notation, hence $n = |V|$ and $m = |E|$, and the minimum degree of G is $\delta(G)$. We describe the efficiency of an exponential-time algorithm using the standard $O^\star$ notation, which suppresses polynomial factors in any parameters.

If $X \subseteq V(G)$, we write $G - X$ to mean the graph formed by deleting the vertices in X (and incident edges) from G. In the case that $X = \{v\}$, we will write $G - v$ rather than $G - \{v\}$.

Treewidth and pathwidth

We first give the definition of treewidth and pathwidth (from [3]).

A tree (resp. path) decomposition of a graph $G = (V, E)$ is a pair $(\{X_i | i \in I\}, T = (I, F))$ where $\{X_i | i \in I\}$ is a family of subsets of V, one for each node of T, and T is a tree (resp. path) such that

1. $\bigcup_{i \in I} X_i = V$;
2. for all edges $vw \in E$, there exists an $i \in I$ with $v, w \in X_i$;
3. for all $i, j, k \in I$, if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The width of a tree (resp. path) decomposition is $\max_{i \in I} |X_i| - 1$. The treewidth (resp. pathwidth) of a graph G is the minimum width over all possible tree (resp. path) decompositions of G.

It is well known that trees (with at least one edge) have treewidth 1, and series-parallel graphs have treewidth at most 2. The pathwidth of these graphs is $O(\log n)$.

Max 2-CSP

An instance of the maximum 2-constraint satisfaction problem (Max 2-CSP) consists of a simple graph $G = (V, E)$, called the *constraint graph*, a set of colours $[r] = \{0, 1, \ldots, r-1\}$, for some $r \geq 2$, and a constant $S_\emptyset$. Additionally, for each edge and vertex of G a score function is supplied, where the score function of an edge uv takes the form $S_{uv} : [r]^2 \to \mathbb{R}$, and the score function of a vertex v is of the form $S_v : [r] \to \mathbb{R}$. Note that only one score is defined for a given colouring of each edge uv, so for colours $c_1, c_2 \in [r]$ we consider $S_{uv}(c_1, c_2)$ and $S_{vu}(c_2, c_1)$ to be equivalent names for the same score.

Any colouring ψ of the vertices of G using the set of colours $[r]$ induces a *cost* which is the sum of the vertex and edge functions plus $S_\emptyset$:

$$S(\psi) = S_\emptyset + \sum_{v \in V} S_v(\psi(v)) + \sum_{uv \in E} S_{uv}(\psi(u), \psi(v))$$

A *candidate solution*, or more simply, a *solution* of a Max 2-CSP instance is any function $\psi : V \to [r]$ which assigns a colour to each vertex of G. An *optimal solution* is a solution which maximizes $S(\psi)$, and the goal of the Max 2-CSP problem is to find an optimal solution.

The set of problems that lie within Max 2-CSP is quite rich; the reader may be interested in verifying that Maximum Cut, Maximum Directed Cut, Maximum Independent Set, Minimum Vertex Cover, and Maximum 2-Sat are all examples of problems that can be modelled as a Max 2-CSP instance.

3 A general reduction result

We now prove a general reduction theorem. The idea is to reduce graphs until the components are of two types, either cubic, or such that deleting a single vertex gives a series-parallel graph. The theorem allows us to use upper bounds for certain graph parameters on these two types of graphs to give upper bounds for all graphs.

Note that by a series-parallel graph we mean here a simple graph with no K_4-minor or, equivalently, a graph of treewidth at most 2.

To state the theorem we need to define several functions.

3.1 Definition and properties of function g_α

We first define the important function g_α, (a generalisation of the function g defined in [8] and the same function in [16]). For $\alpha = \frac{1}{4}, \frac{1}{6}$, the values of these functions, mostly for small arguments, have appeared in a number of papers [8, 10, 14, 15, 16, 17, 21, 28], however no attempt appears to have been made previously to analyse the behaviour of the function or to generalise the reduction techniques used. The function g_α acts as a cost function applied to the degree of each vertex. The motivation for the definition is roughly as follows: In a reduction and branching algorithm, it appears that the best results often arise from deleting a vertex w of maximum degree Δ with a neighbour v_1 of lower degree. This results in a reduction in the total cost of the vertices of at least the sum of (i) $g_\alpha(\Delta)$, the contribution from w itself, (ii) $g_\alpha(\Delta-1)-g_\alpha(\Delta-2)$, the decrease in the contribution from v_1, and (iii) $(\Delta-1)(g_\alpha(\Delta)-g_\alpha(\Delta-1))$, the decrease in the contributions from the other neighbours of w. Thus the total reduction is $\Delta g_\alpha(\Delta)-(\Delta-2)g_\alpha(\Delta-1)-g_\alpha(\Delta-2)$. The function g_α is defined precisely to make this sum 1, which is the cost of deleting a vertex.

Definition of functions g_α and g'_α

For any $n \geq 2$, and α with $0 \leq \alpha \leq 1$, define the function $g_\alpha(n)$ by setting $g_\alpha(2) = 0$, $g_\alpha(3) = \alpha$, and for any $n \geq 4$,

$$ng_\alpha(n) = (n-2)g_\alpha(n-1) + g_\alpha(n-2) + 1.$$

We extend g_α to all real numbers at least 2 by linear interpolation, i.e., if $r = n+x$, where $n \geq 2$ is an integer, and $0 \leq x \leq 1$, then we set $g_\alpha(r) = (1-x)g_\alpha(n) + xg_\alpha(n+1)$.

Also, for any $n \geq 3$, define $g'_\alpha(n) = g_\alpha(n) - g_\alpha(n-1)$.

Properties of g_α and g'_α

We now list the properties of g_α and g'_α which we will use later. We defer proofs of these properties to the Appendix (Section 8); see Lemmas 8.1 to 8.5.

1. For any integer $n \geq 2$,

$$g_\alpha(n) = (4-3\alpha)\frac{A(n)}{n!} + (2-3\alpha)\frac{(-1)^n}{n!} - (3-3\alpha)$$

   where $A(n)$ is the alternating factorial function given by

$$A(n) = n! - (n-1)! + \ldots - (-1)^n \cdot 1!.$$

2. For all real $d \geq 2$,

$$g_\alpha(d) = 1 - \frac{4-3\alpha}{d+1} + O(1/d^3).$$

3. If $\alpha \leq 1/2$, then g_α is non-decreasing, i.e.,

$$g_\alpha(n+1) \geq g_\alpha(n)$$

   for all integers $n \geq 2$.

4. If $1/6 \leq \alpha \leq 3/10$, then g'_α is non-increasing, i.e.,

$$g'_\alpha(n+1) \leq g'_\alpha(n)$$

for all integers $n \geq 3$.

5. If $1/6 \leq \alpha \leq 3/10$, $g_\alpha(n)/n$ is strictly decreasing for $n \geq 5$.

6. For all $n \geq 4$,

$$1 - g_\alpha(n) = (n-1)g'_\alpha(n) + g'_\alpha(n-1).$$

This is just a rearrangement of the defining recurrence.

We list the first few values of $g_{1/4}$ and $g_{1/6}$ (which we will use later in the paper) below:

d	2	3	4	5	6	7	8
$g_{1/4}(d)$	0	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{19}{40}$	$\frac{131}{240}$	$\frac{1009}{1680}$	$\frac{8651}{13440}$
$g_{1/6}(d)$	0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{13}{30}$	$\frac{23}{45}$	$\frac{359}{630}$	$\frac{1553}{2520}$

For α outside the range $[1/6, 3/10]$, the function g'_α is not monotone, and the reduction method used below breaks down. It is possible that one could redefine g_α in these cases and prove a similar reduction result, however since all the cases of interest have $1/6 \leq \alpha \leq 3/10$, we do not pursue that here.

Definitions of functions f_α, f^-_α

We now define the functions which will be used as upper bounds. For any graph G with minimum degree at least 2, define $f_\alpha(G)$ by:

$$f_\alpha(G) = \sum_{v \in V(G)} g_\alpha(d_G(v)).$$

Although $f_\alpha(G)$ is the main upper bound of interest, the inductive proof relies crucially on a small negative term, which depends on the minimum degree of each component. If G has minimum degree at least 3, with connected components $G_1, \ldots, G_k$, set

$$f^-_\alpha(G) = f_\alpha(G) - \sum_{i=1}^{k} g'_\alpha(\delta(G_i)) \quad = \sum_{i=1}^{k} f^-_\alpha(G_i).$$

Note that despite the negative term, $f^-_\alpha(G)$ is always strictly positive.

3.2 Reduction

We also need the concept of series-parallel reductions. Let G be a graph, and consider the following four operations on G:

1. Delete an isolated vertex of G.
2. Delete a vertex of degree 1 (and its incident edge).
3. Let v be a vertex of degree 2 with non-adjacent neighbours x and y; delete v (and edges vx, vy) and add an edge between x and y.
4. Let v be a vertex of degree 2 with adjacent neighbours; delete v (and incident edges).

Let $r(G)$ be a graph obtained from G by applying operations $1, 2, 3, 4$ above repeatedly until none is possible (because the graph has minimum degree at least 3 or is empty). It follows from the definition of the series-parallel property that G is series-parallel if and only if $r(G)$ is empty. These reductions have been used by many authors, for example [5, 11, 14, 28]. (The resulting graph $r(G)$ is in fact unique.)

Definition of basic graph

We will say a connected graph of minimum degree at least 3 is basic if either G is cubic or $r(G - v)$ is empty for some $v \in V$. A graph is basic if every component is basic.

Reduction-deletion order

We will require the following lemma, which says that the order of deleting vertices and reducing does not change the graph that results.

Lemma 3.1 *Let $G = (V, E)$ be a graph, w be a vertex of G, and X be a subset of $V(r(G - w))$. Then*

$$r(G - (X \cup \{w\})) = r(r(G - w) - X).$$

Proof. This follows from the results of Kneis *et al.* [21]. For a subset D of V, they define a valid reduction sequence to be a sequence of operations, each of which is either a series-parallel reduction of a vertex not in D, or removal of a vertex in D. They prove that any two maximal valid reduction sequences produce the same graph.

Let $D = X \cup \{w\}$. Then we have two maximal valid reduction sequences as follows:

(a) remove each element of $X \cup \{w\}$ in turn, then apply series-parallel reductions as long as possible. This gives $r(G - (X \cup \{w\}))$.

(b) remove w, then apply the sequence of series-parallel reductions which produces $r(G - w)$. Since every vertex of X is present in $r(G - w)$, none of these series-parallel reductions involves a vertex in D, so the sequence is valid. Then remove each element of X in turn, and apply series-parallel reductions as long as possible. This gives $r(r(G - w) - X)$.

These two sequences must give the same graph, so $r(G-(X\cup\{w\})) = r(r(G-w)-X)$. □

3.3 The main theorem

We now prove the main theorem which allows us to extend results on basic graphs to general graphs. The essential idea of the theorem is quite simple; we have some graph parameter p which is (more or less) invariant under series-parallel reduction and satisfies $p(G) \leq 1 + p(G-w)$ for any vertex w, and we have bounds of the form $f_\alpha(G)$ for the parameter on basic graphs. We want a bound (also of the form $f_\alpha(G)$) on the parameter for general graphs.

If the graph is connected and not regular, then we can delete a vertex w of maximum degree Δ with at least one neighbour v_1 of lower degree, and then reduce $G-w$ to obtain G^*, then the value of f_α will decrease by at least the sum of (i) $g_\alpha(\Delta)$, the contribution from w itself, (ii) $g'_\alpha(\Delta-1)$, the decrease in the contribution from v_1, and (iii) $(\Delta-1)g'_\alpha(\Delta)$, the decrease in the contributions from the other neighbours of w. The function g_α is defined precisely to make this sum 1, and by induction $p(G-w) = p(G^*) \leq f_\alpha(G^*) \leq f_\alpha(G)-1$, so $p(G) \leq p(G-w)+1 \leq f_\alpha(G)$.

However we also have to deal with the case when the graph is regular. For cubic graphs this is covered by the bounds for basic graphs. Otherwise, this is addressed by using $f^-_\alpha(G)$ rather than $f_\alpha(G)$ as the upper bound, giving a slightly stronger inductive hypothesis. However we then have to take care that the inductive step works in all cases, in particular when w is a cut-vertex or when the minimum degree changes, and much of the proof is devoted to this.

Theorem 3.2 *Let $G=(V,E)$ be a graph with n vertices and minimum degree $\delta \geq 3$. Then there is a set $X_G \subseteq V$ such that (i) $r(G-X_G)$ is non-empty and basic, and (ii) $|X_G| + f^-_\alpha(r(G-X_G)) \leq f^-_\alpha(G)$ for any α, $1/6 \leq \alpha \leq 3/10$.*

Proof. We use induction on n, the number of vertices of G. If $n<4$, there is nothing to prove, so assume that $n \geq 4$, and the theorem holds for all G with fewer than n vertices. If G is basic, then set $X_G = \emptyset$. Since $|X_G| = 0$ and $r(G-X_G) = G$, the result follows. So assume that G is not basic. If G is not connected, then suppose that G has connected components $G_1,\ldots,G_k$, where $k \geq 2$. By the inductive hypothesis, for each i there is a set X_{G_i} such that $r(G_i - X_{G_i})$ is non-empty and basic and such that

$$|X_{G_i}| + f^-_\alpha(r(G_i - X_{G_i})) \leq f^-_\alpha(G_i)$$

Let $X_G = \bigcup_{i=1}^k X_{G_i}$. Then it is clear that $r(G-X_G) = \bigcup_{i=1}^k r(G_i - X_{G_i})$, so $r(G-X_G)$ is non-empty and basic. Also $f^-_\alpha(r(G-X_G)) = \sum_{i=1}^k f^-_\alpha(r(G_i-X_{G_i}))$. Thus since $|X_G| = \sum_{i=1}^k |X_{G_i}|$, we have

$$|X_G| + f^-_\alpha(r(G-X_G)) = \sum_{i=1}^k \left(|X_{G_i}| + f^-_\alpha(r(G_i-X_{G_i}))\right) \leq \sum_{i=1}^k f^-_\alpha(G_i) = f^-_\alpha(G),$$

as required.

So we may assume that G is connected and not basic. Let Δ be the maximum degree of G. Since G is not basic, we have $\Delta > 3$.

Let w be a vertex of maximum degree in G, chosen so that the degree of its lowest degree neighbour is as small as possible. Let the neighbours of w be $v_1, v_2, \ldots, v_\Delta$, where $d_G(v_1) \leq d_G(v_2) \leq \ldots \leq d_G(v_\Delta)$. Since G is connected, $d_G(v_1) < \Delta$ unless G is Δ-regular.

Delete the vertex w to form the graph $G' = G - w$, and let G^* be the reduced graph $r(G')$. Since we are assuming that G is not basic, G^* is not empty.

By the inductive hypothesis, there is a set X_{G^*} such that $r(G^* - X_{G^*})$ is non-empty and basic and such that

$$|X_{G^*}| + f_\alpha^-(r(G^* - X_{G^*})) \leq f_\alpha^-(G^*).$$

Set $X_G = X_{G^*} \cup \{w\}$. Now

$$r(G^* - X_{G^*}) = r(r(G - w) - X_{G^*}) = r(G - (X_{G^*} \cup \{w\})) = r(G - X_G),$$

by Lemma 3.1. Thus $r(G - X_G)$ is non-empty and basic. Also

$$|X_G| + f_\alpha^-(r(G - X_G)) = 1 + |X_{G^*}| + f_\alpha^-(r(G^* - X_{G^*})) \leq 1 + f_\alpha^-(G^*).$$

Thus it suffices to show that

$$1 + f_\alpha^-(G^*) \leq f_\alpha^-(G) = f_\alpha(G) - g'_\alpha(\delta(G)). \tag{1}$$

The rest of the proof is devoted to showing this. First note that

$$\begin{aligned} 1 + f_\alpha^-(G^*) &\leq 1 + f_\alpha(G^*) - g'_\alpha(\delta(G^*)) \\ &= 1 + f_\alpha(G') - (f_\alpha(G') - f_\alpha(G^*) + g'_\alpha(\delta(G^*))). \end{aligned} \tag{2}$$

Now

$$\begin{aligned} 1 + f_\alpha(G') &= 1 + \sum_{v \in V(G')} g_\alpha(d_{G'}(v)) \\ &= 1 + \sum_{v \in V(G') \setminus \{v_1, \ldots, v_\Delta\}} g_\alpha(d_{G'}(v)) + \sum_{i=1}^{\Delta} g_\alpha(d_{G'}(v_i)) \\ &= 1 + \sum_{v \in V(G') \setminus \{v_1, \ldots, v_\Delta\}} g_\alpha(d_G(v)) + \sum_{i=1}^{\Delta} g_\alpha(d_G(v_i) - 1) \\ &= 1 + \sum_{v \in V(G')} g_\alpha(d_G(v)) + \sum_{i=1}^{\Delta} (g_\alpha(d_G(v_i) - 1) - g_\alpha(d_G(v_i))) \\ &= 1 + \sum_{v \in V(G)} g_\alpha(d_G(v)) - g_\alpha(d_G(w)) - \sum_{i=1}^{\Delta} g'_\alpha(d_G(v_i)) \\ &= 1 + f_\alpha(G) - g_\alpha(\Delta) - \sum_{i=1}^{\Delta} g'_\alpha(d_G(v_i)) \\ &= f_\alpha(G) + (\Delta - 1) g'_\alpha(\Delta) + g'_\alpha(\Delta - 1) - \sum_{i=1}^{\Delta} g'_\alpha(d_G(v_i)) \end{aligned} \tag{3}$$

$$
\begin{aligned}
&= f_\alpha(G) - (g'_\alpha(d_G(v_1)) - g'_\alpha(\Delta-1)) - (g'_\alpha(d_G(v_2)) - g'_\alpha(\Delta)) \\
&\quad - \sum_{i=3}^{\Delta} (g'_\alpha(d_G(v_i)) - g'_\alpha(\Delta)) \qquad (4)
\end{aligned}
$$

where to obtain (3) we use property 6 of g_α, that $1 - g_\alpha(\Delta) = (\Delta-1)g'_\alpha(\Delta) + g'_\alpha(\Delta-1)$.

Thus, substituting expression (4) for $1 + f_\alpha(G')$ in (2), we have

$$
\begin{aligned}
1 + f^-_\alpha(G^*) \;\leq\; & f_\alpha(G) - (g'_\alpha(d_G(v_1)) - g'_\alpha(\Delta-1)) - (g'_\alpha(d_G(v_2)) - g'_\alpha(\Delta)) \\
& - \sum_{i=3}^{\Delta} (g'_\alpha(d_G(v_i)) - g'_\alpha(\Delta)) \\
& - (f_\alpha(G') - f_\alpha(G^*) + g'_\alpha(\delta(G^*))). \qquad (5)
\end{aligned}
$$

Let

$$
\begin{aligned}
j_1(G) &= g'_\alpha(d_G(v_1)) - g'_\alpha(\Delta-1) \\
j_2(G) &= g'_\alpha(d_G(v_2)) - g'_\alpha(\Delta) \\
j_3(G) &= \sum_{i=3}^{\Delta} (g'_\alpha(d_G(v_i)) - g'_\alpha(\Delta)) \\
j_4(G) &= f_\alpha(G') - f_\alpha(G^*) + g'_\alpha(\delta(G^*)).
\end{aligned}
$$

Note that only $j_1(G)$ can be strictly negative, and then only in the case that $d_G(v_1) = \Delta$.

Now (5) becomes

$$1 + f^-_\alpha(G^*) \leq f_\alpha(G) - j_1(G) - j_2(G) - j_3(G) - j_4(G).$$

and recall, from (1), that we need to show that

$$1 + f^-_\alpha(G^*) \leq f_\alpha(G) - g'_\alpha(\delta(G)).$$

Thus it suffices to establish that, in all cases,

$$j(G) = j_1(G) + j_2(G) + j_3(G) + j_4(G) \geq g'_\alpha(\delta(G)).$$

To prove this, we first note that if G' contains a vertex v of degree $d \geq 3$, then

$$j_4(G) = f_\alpha(G') - f_\alpha(G^*) + g'_\alpha(\delta(G^*)) \geq g'_\alpha(d).$$

For either (i) the vertex v is present (perhaps with reduced degree) in G^*, so that $\delta(G^*) \leq d$ and so, since g'_α is non-increasing, $g'_\alpha(\delta(G^*)) \geq g'_\alpha(d)$, or (ii) v is in G' but not in G^*, which implies that $f_\alpha(G') - f_\alpha(G^*) \geq g_\alpha(d_{G'}(v)) = g_\alpha(d) \geq g'_\alpha(d)$, as required.

If G is Δ-regular, then $d_G(v_1) = \Delta$, and G' will contain a vertex of degree $\Delta - 1$, so that $j_4(G) \geq g'_\alpha(\Delta-1)$ and $j(G) \geq j_1(G) + j_4(G) \geq (g'_\alpha(d_G(v_1)) - g'_\alpha(\Delta-1)) + g'_\alpha(\Delta-1) = g'_\alpha(\Delta) = g'_\alpha(\delta(G))$, as required.

Otherwise, we have $d_G(v_1) \leq \Delta - 1$, so that $j_1(G) \geq 0$. Hence from now on, $j_i(G) \geq 0$ in all cases. If G' contains a vertex v with $\delta(G) \geq d_{G'}(v) \geq 3$, then we have $j(G) \geq j_4(G) \geq g'_\alpha(d_{G'}(v)) \geq g'_\alpha(\delta(G))$, as required.

So suppose that G' does not contain a vertex v with $\delta(G) \geq d_{G'}(v) \geq 3$. Then we must have $\delta(G) = 3$, for if $\delta(G) > 3$, then we can choose a vertex $v \neq w$ with $d_G(v) = \delta(G)$, and then v will have degree $\delta(G)$ or $\delta(G) - 1$ in G' and in either case will satisfy $\delta(G) \geq d_{G'}(v) \geq 3$, a contradiction. Also, every vertex with $d_G(v) = 3$ must be a neighbour of w, for otherwise such a vertex would satisfy $\delta(G) \geq d_{G'}(v) \geq 3$.

Thus we may assume that $\delta(G) = 3$ and every vertex with $d_G(v) = 3$ is a neighbour of w. If $d_G(v_\Delta) > 3$, then vertex v_Δ satisfies $\Delta - 1 \geq d_{G'}(v_\Delta) \geq 3$, so $j_4(G) \geq g'_\alpha(d_{G'}(v_\Delta)) \geq g'_\alpha(\Delta - 1)$. Also $d_G(v_1) = 3$ so $j_1(G) = g'_\alpha(3) - g'_\alpha(\Delta - 1)$, so we have

$$j(G) \geq j_1(G) + j_4(G) \geq g'_\alpha(3) - g'_\alpha(\Delta - 1) + g'_\alpha(\Delta - 1) = g'_\alpha(3) = g'_\alpha(\delta(G))$$

as required.

The only other possibility is that $d_G(v_1) = d_G(v_2) = \ldots = d_G(v_\Delta) = 3$. Hence all of $v_1, \ldots, v_\Delta$ have degree 2 in G' and so are removed by the reduction to G^*. Thus since we are assuming that G^* is not empty, G' must have some other vertex v not adjacent to w in G, so $d_{G'}(v) \geq 3$. Then we have

$$j(G) \geq j_2(G) + j_4(G) \geq (g'_\alpha(d_G(v_2)) - g'_\alpha(\Delta)) + g'_\alpha(d_{G'}(v)) \geq g'_\alpha(3) = g'_\alpha(\delta(G)),$$

and the result holds. □

Remarks

(i) Note that the set X_G is not necessarily unique.
(ii) Although we have proved the existence of X_G inductively, it is clear that X_G can be found by the following algorithm:

Algorithm X

Input: Graph G with minimum degree at least 3.
1: $X_G \leftarrow \emptyset$
2: **while** G is not basic **do**
3: Choose a component G_i which is not basic
4: Choose a vertex w of maximum degree in G_i, if possible with a neighbour of lower degree
5: $G \leftarrow r(G - w)$
6: $X_G \leftarrow X_G \cup \{w\}$
7: **end while**
Output: X_G.

4 Faster exponential-time algorithms

In this section we will show how the results of Theorems 4.1–4.3 below allow the construction of efficient algorithms to solve Max 2-CSP. The basic idea is to find a subset X of the vertices of the constraint graph whose deletion leaves a (large) graph on the

remaining vertex set Y which has a simple structure, in this case bounded treewidth. Next, we enumerate all possible r-colourings of the set X. Each such colouring c_X is a *partial solution* for G in the sense that only Y remains to be coloured. The next step involves finding an *optimal extension* for c_X, i.e., an optimal colouring for Y subject to c_X. We give details below.

Consider an instance Π of Max 2-CSP with graph $G = (V, E)$, set of colours $[r]$, constant $S_\emptyset$, and functions S_v for each $v \in V$ and S_{uv} for each edge $uv \in E$.

Let $X \subseteq V$ be a subset of the vertices of G, and let $Y = V \setminus X$. Let $G(Y)$ be the subgraph induced by the set Y of vertices. Also let $c : X \to [r]$ be a (not-necessarily proper) r-colouring of the vertices in X.

We construct an instance $\Pi(Y, c)$ of Max 2-CSP on the graph $G(Y)$ as follows:

1. The constant $S_\emptyset^{Y,c}$ is defined by

$$S_\emptyset^{Y,c} = S_\emptyset + \sum_{v \in X} S_v(c(v)) + \sum_{uv \in E(X)} S_{uv}(c(u), c(v));$$

2. for each $v \in Y$, and $i \in [r]$

$$S_v^{Y,c}(i) = S_v(i) + \sum_{w \in X : vw \in E} S_{vw}(i, c(w));$$

3. for each $uv \in E(Y)$, and $i, j \in [r]$,

$$S_{uv}^{Y,c}(i, j) = S_{uv}(i, j).$$

Now it is easy to see that for a fixed set $Y \subseteq V$,

$$\begin{aligned}
\max_{\psi : V \to [r]} S_\Pi(\psi) &= \max_{\psi : V \to [r]} \left[S_\emptyset + \sum_{v \in V} S_v(\psi(v)) + \sum_{uv \in E} S_{uv}(\psi(u), \psi(v)) \right] \\
&= \max_{c : X \to [r]} \max_{\psi^Y : Y \to [r]} \left[S_\emptyset^{Y,c} + \sum_{v \in V} S_v^{Y,c}(\psi^Y(v)) + \sum_{uv \in E} S_{uv}^{Y,c}(\psi^Y(u), \psi^Y(v)) \right] \\
&= \max_{c : X \to [r]} S_{\Pi(Y,c)}(\psi^Y).
\end{aligned}$$

Thus by trying every possible colouring c of X, and solving the corresponding instance $\Pi(Y, c)$, we can solve Π in $r^{|X|}$ times the time taken to solve each sub-instance $\Pi(Y, c)$. If we can choose Y so that these sub-instances can be solved in polynomial time, then the instance Π can be solved in total time $O^\star(r^{|X|})$, and polynomial space.

Let W_5 be the wheel on 5 vertices, and let $w_5(G)$ be the smallest size of a set $X \subseteq V(G)$ such that $G - X$ has no W_5-minor. It is easy to see that for any graph G, we have $w_5(G) = w_5(r(G))$ where $r(G)$ is the reduced graph defined above.

Edwards and Farr [8] prove the following:

Theorem 4.1 ([8]) *Let $G = (V, E)$ be a connected graph with n vertices and minimum degree $\delta \geq 3$. Then*

$$w_5(G) \leq \sum_{v \in V(G)} g_{1/4}(d(v)) \; - g'_{1/4}(\delta) = f^-_{1/4}(G).$$

□

We briefly show how this result can be derived from the general reduction theorem 3.2 above. We first show that for any basic graph G, $w_5(G) \leq f^-_{1/4}(G)$. To see this, note that if G has connected components $G_1, \ldots, G_k$, then $w_5(G) = \sum_{i=1}^k w_5(G_i)$ and $f^-_{1/4}(G) = \sum_{i=1}^k f^-_{1/4}(G_i)$, so we may assume that G is connected. If G is cubic, then it was shown in [8] that $w_5(G) \leq (n-1)/4 = f^-_{1/4}(G)$. Otherwise $\Delta(G) > 3$ and $r(G - v)$ is empty for some v. Then $G - v$ is series-parallel and so certainly W_5-minor-free, and $w_5(G) \leq 1$. But $f^-_{1/4}(G) \geq g_{1/4}(4) + 4g_{1/4}(3) - g'_{1/4}(3) > 1$, as required.

Next, by Theorem 3.2, there is a set X_G such that

$$|X_G| + f^-_{1/4}(r(G - X_G)) \leq f^-_{1/4}(G),$$

and $r(G - X_G)$ is basic and non-empty. Then it is clear that $w_5(G) \leq |X_G| + w_5(G - X_G)$, and since $w_5(G - X_G) = w_5(r(G - X_G))$, we have

$$\begin{aligned} w_5(G) &\leq |X_G| + w_5(G - X_G) \\ &= |X_G| + w_5(r(G - X_G)) \\ &\leq |X_G| + f^-_{1/4}(r(G - X_G)) \\ &\leq f^-_{1/4}(G), \end{aligned}$$

as required.

As corollaries, we obtain the following.

Theorem 4.2 ([8]) *Let G be a graph, and $r(G)$ the reduced graph of G. Then*

$$w_5(G) \leq \sum_{v \in V(r(G))} g_{1/4}(d_{r(G)}(v)).$$

□

Theorem 4.3 ([8]) *Let G be a graph with n vertices, of average degree $d \geq 2$. Then if G is connected, or $d \geq 5$,*

$$w_5(G) \leq g_{1/4}(d)\, n.$$

□

We note that since W_5 is planar, the W_5-minor-free induced subgraphs obtained above have bounded treewidth. In fact since W_5 is a minor of each of the four graphs in the forbidden minor characterisation of treewidth 3 graphs given by Arnborg *et al.* [1], then the subgraphs found have treewidth at most 3. It is well-known that Max 2-CSP on graphs of bounded treewidth can be solved in polynomial time; see for example [29]. Then we have the following theorem.

Theorem 4.4 *Let G be a graph, and $r(G)$ the reduced graph of G. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{\beta(G)})$$

where $\beta(G) = \sum_{v \in V(r(G))} g_{1/4}(d_{r(G)}(v))$.

Proof. We use the following algorithm (Algorithm E). Let Π be an instance of Max 2-CSP, with constraint graph G. We first find the reduced graph $r(G)$, then, using Algorithm X and the results of [8], it is easy to find a set X of size at most $\beta(G)$ such that $G - X$ has treewidth at most 3. Then as above, we try all possible colourings of X, and, for each colouring c, solve the instance $\Pi(Y, c)$ in polynomial time. Then the algorithm uses time $O^\star(r^{|X|})$ and polynomial space. □

Although the analysis splits the finding of the set X into two distinct stages, we can combine these to produce the following:

Algorithm E
Input: $\Pi = (G = (V, E), r, S_\emptyset, \{S_v | v \in V\}, \{S_{uv} | uv \in E\})$.
1: $G \leftarrow r(G)$
2: $X \leftarrow \emptyset$
3: **while** some component of G_i of G has at least 5 vertices **do**
4: Choose a vertex w of maximum degree in G_i,
if possible with a neighbour of lower degree
5: $G \leftarrow r(G - w)$
6: $X \leftarrow X \cup \{w\}$
7: **end while**
8: $Y \leftarrow V \setminus X$.
9: **for each** colouring $c : X \to \{1, \ldots, r\}$ **do**
10: Solve $\Pi(Y, c)$
11: **end for**
Output: Best solution found

Note that $G - X$ will not necessarily be series-parallel, for example when $G = K_4$ (so $X = \emptyset$). However $G - X$ will always have treewidth at most 3. We also observe that in practice better results will be achieved if (a) the algorithm solves the problem on components separately whenever the graph splits, and (b) reduction stops as soon as graph of treewidth 3 is obtained. However it is not clear that either of these will lead to an improvement in the time complexity below.

Theorem 4.5 *Let G be a connected graph with n vertices, of average degree $d \geq 2$. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{g_{1/4}(d)n}) = O^\star(r^{(1 - \frac{13/4}{d+1} + O(1/d^3))n}).$$

□

This improves on the previous best exponent of $(1 - \frac{3}{d+1})n$ obtained by Golovnev and Kutzkov [17] (Algorithm D).

Re-expressing this bound in terms of the number m of edges, we have, since $2m = dn$,

Theorem 4.6 *Let G be a connected graph with m edges, of average degree $d \geq 2$. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{(2g_{1/4}(d)/d)m}).$$

□

Since $g_{1/4}(d)/d$ is strictly decreasing for $d \geq 5$ (Section 3.1, property 5), it follows that the maximum value of the fraction $2g_{1/4}(d)/d$ occurs when $d = 5$, when $g_{1/4}(d) = 19/40$ and so $2g_{1/4}(d)/d = 19/100$. Hence the exponent in this case is the same as that obtained by Scott and Sorkin [28], but for all other d, it is lower. For example, for a graph of average degree 10, Scott and Sorkin's Algorithm B has time bound $O^\star(r^{0.95n})$ and Golovnev and Kutzkov's Algorithm D is $O^\star(r^{0.728n})$, whereas our Algorithm E is $O^\star(r^{0.708n})$ approximately.

There are three main differences between our Algorithm E and the earlier Algorithms B and D: (i) Algorithm E treats all degrees at least 4 in a uniform way, unlike Algorithm B, (ii) Algorithm E always picks a vertex of maximum degree with a lower degree neighbour if possible, unlike algorithm D; (iii) Algorithms B and D essentially find an induced subgraph (of the constraint graph) which is series-parallel (treewidth 2), whereas Algorithm E finds an induced subgraph of treewidth at most 3. This allows a (very) slightly smaller number of deletions in one of the base cases (connected cubic graphs). This in turn allows a slight strengthening of the inductive hypothesis which enables the induction to proceed.

Our Algorithm E achieves both the best bound in terms of the average degree and the best bound expressed in terms of edges. We will discuss in Section 5 how much further improvement might be possible.

We have presented Algorithm E as a two-stage process in which we first find a vertex subset X whose deletion leaves a graph of treewidth at most 3, and then (for each colouring of this deleted set) determine the optimum extension to the low treewidth subgraph. It should be clear that it is also possible to implement essentially the same algorithm in the manner used by a number of earlier algorithms, in which we alternately delete one of the vertices in X and then reduce the resulting graph. Each deletion requires a branching of the algorithm (with one branch for each possible colour of the deleted vertex), while the reductions simply modify the CSP instance on the remaining graph.

Extension to polynomial 2-CSP

Scott and Sorkin [29] define a wider class of problems called polynomial 2-CSP, and show that several existing algorithms, including their Algorithm B, can be extended to deal with these problems. From the remarks above, it is clear that our Algorithm E can be extended in the same way.

4.1 Max 2-CSP for claw-free graphs

Cheng *et al.* [4] show that a cubic claw-free graph (i.e., one with no induced $K_{1,3}$ subgraph) with n vertices can be made planar by deleting at most $n/6$ vertices. In fact it is easy to

see that the resulting graph also has treewidth at most 3 (though the paper only claims treewidth at most 4). Furthermore, the bound of $n/6$ derives from an algorithm in which deletion of a vertex always results in the reduction of at least five further vertices. However the last of these deletions is unnecessary, as when $n \leq 6$ the graph already has treewidth at most 3 (and is planar). It follows that a graph of treewidth at most 3 can be obtained from a connected cubic claw-free graph by deleting at most $(n-1)/6$ of its vertices.

Let $t_3(G)$ be the smallest size of a set $X \subseteq V(G)$ such that $G-X$ has treewidth at most 3. Then we have the following theorem:

Theorem 4.7 *Let $G = (V, E)$ be a claw-free graph with n vertices and minimum degree $\delta \geq 3$. Then*

$$t_3(G) \leq f^-_{1/6}(G).$$

Proof. We first show that for any basic claw-free graph G, $t_3(G) \leq f^-_{1/6}(G)$. To see this, note that if G has connected components $G_1, \ldots, G_k$, then $t_3(G) = \sum_{i=1}^k t_3(G_i)$ and $f^-_{1/6}(G) = \sum_{i=1}^k f^-_{1/6}(G_i)$, so we may assume that G is connected. If G is cubic, then it follows from the results of Cheng *et al.* above that $t_3(G) \leq (n-1)/6 = f^-_{1/6}(G)$. Otherwise $\Delta(G) > 3$ and $r(G-v)$ is empty for some v. Then $G-v$ is series-parallel and $\mathrm{tw}(G-v) \leq 2$, so $\mathrm{tw}(G) \leq 3$. Hence $t_3(G) = 0 \leq f^-_{1/6}(G)$.

Next, by Theorem 3.2, there is a set X_G such that

$$|X_G| + f^-_{1/6}(r(G - X_G)) \leq f^-_{1/6}(G),$$

and $r(G - X_G)$ is basic and non-empty. Also, since G is claw-free, it is easy to see that $r(G - X_G)$ is also. Thus $t_3(r(G - X_G)) \leq f^-_{1/6}(r(G - X_G))$ from above. Then it is clear that $t_3(G) \leq |X_G| + t_3(G - X_G)$, and $t_3(G - X_G) = t_3(r(G - X_G))$, so we have

$$\begin{aligned} t_3(G) &\leq |X_G| + t_3(G - X_G) \\ &= |X_G| + t_3(r(G - X_G)) \\ &\leq |X_G| + f^-_{1/6}(r(G - X_G)) \\ &\leq f^-_{1/6}(G), \end{aligned}$$

as required.

Remark

An almost identical proof shows that we can obtain a W_6-minor-free graph by deleting the same number of vertices. Let $w_6(G)$ be the smallest size of a set $X \subseteq V(G)$ such that $G - X$ has no W_6-minor. Then the only change required in order to show that $w_6(G) \leq f^-_{1/6}(G)$ for any claw-free graph is the following: Whereas $t_3(G) = 0$ whenever $r(G-v)$ is empty, this is not necessarily true for $w_6(G)$. However we do have $w_6(G) \leq 1$, and if $n \geq 6$, then $f^-_{1/6}(G) \geq g_{1/6}(4) + 5g_{1/6}(3) - g'_{1/6}(3) = 1$, as required. If $n \leq 5$, then G cannot have a W_6-minor, so $w_6(G) = 0 \leq f^-_{1/6}(G)$.

As before, we have the following corollaries:

Theorem 4.8 *Let G be a claw-free graph, and $r(G)$ the reduced graph of G. Then*

$$t_3(G) \leq \sum_{v \in V(r(G))} g_{1/6}(d_{r(G)}(v)).$$

□

Theorem 4.9 *Let G be a claw-free graph with n vertices, of average degree $d \geq 2$. Then if G is connected, or $d \geq 5$,*

$$t_3(G) \leq g_{1/6}(d)\, n.$$

Proof. As for $g_{1/4}$ in [8], it can be shown that if G is a graph with n vertices and average degree at most d, where $d \geq 2$, and if (i) G is connected or (ii) $d \geq 5$, then we have

$$f_\alpha(r(G)) = \sum_{v \in V(r(G))} g_\alpha(d_{r(G)}(v)) \leq g_\alpha(d)n$$

provided $1/6 \leq \alpha \leq 3/10$, and the result follows. □

Since the subgraphs obtained have treewidth at most 3 then, as before, instances of Max 2-CSP on these subgraphs can be solved in polynomial time. Thus we have

Theorem 4.10 *Let G be a claw-free graph, and $r(G)$ the reduced graph of G. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{\beta(G)}),$$

where $\beta(G) = \sum_{v \in V(r(G))} g_{1/6}(d_{r(G)}(v))$. □

We also have

Theorem 4.11 *Let G be a connected claw-free graph of average degree $d \geq 2$, with n vertices and m edges. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{g_{1/6}(d)n}) = O^\star(r^{(2g_{1/6}(d)/d)m}) = O^\star(r^{(1-\frac{7/2}{d+1}+O(1/d^3))n}).$$

□

We know that $g_{1/6}(d)/d$ is strictly decreasing for all $d \geq 5$ (by property 5 of Section 3.1). By inspection of the values for small d we find that the maximum of $2g_{1/6}(d)/d$ again occurs when $d = 5$. We have $g_{1/6}(5) = 13/30$ so $2g_{1/6}(5)/5 = 13/75$. Thus in the worst case, instances of Max 2-CSP on claw-free graphs can be solved in time $O^\star(r^{(13/75)m})$ and polynomial space.

5 Pathwidth

Scott and Sorkin [28] derive a general upper bound of $(13/75 + o(1))m$ on the treewidth of a graph with m edges and use this to give a further algorithm (C) for Max 2-CSP with improved time complexity of $O^\star(r^{(13/75+o(1))m})$, but also requiring exponential space. Kneis *et al.* use similar methods to obtain an upper bound of $(13/75 + o(1))m + O(\log n)$ on the pathwidth of a graph. Fomin *et al.* [10] show that

$$\mathrm{pw}(G) \leq \tfrac{1}{6}n_3 + \tfrac{1}{3}n_4 + \tfrac{13}{30}n_5 + \tfrac{23}{45}n_6 + n_{\geq 7} + o(n) = \textstyle\sum_{d=3}^{6} g_{1/6}(d)n_d + n_{\geq 7} + o(n),$$

where n_i is the number of vertices of degree i, and $n_{\geq 7}$ is the number of vertices of degree at least 7. Gaspers [14] states that this can be extended to a similar bound

$$\mathrm{pw}(G) \leq \sum_{d=3}^{17} \beta_d n_d + n_{\geq 18} + o(n),$$

where the coefficients β_d for $d \in \{7, \ldots, 17\}$ were found by a computer search. It is clear from the table of values given that each constant β_d is an approximation to $g_{1/6}(d)$.

We remove the error terms $n_{\geq 7}$, $n_{\geq 18}$ and show below that

$$\mathrm{pw}(G) \leq \sum_{d \geq 3} g_{1/6}(d)n_d + o(n).$$

We proceed as follows. Fomin and Høie [12] showed that a 3-regular graph G with m edges has pathwidth at most $(1/9 + o(1))m + O(\log n) = (1/6 + o(1))n + O(\log n)$. Hence for 3-regular graphs, there is a function τ, with $\tau(n) = o(n)$, such that $\mathrm{pw}(G) \leq n/6 - 1/6 + \tau(n) + O(\log n)$ (the negative term is needed below). We can also assume that τ is an increasing function of n (otherwise replace τ with σ given by $\sigma(n) = \max_{i \leq n} \tau(i)$).

Then we have the following:

Lemma 5.1 *Let $G = (V, E)$ be a graph with n vertices and minimum degree $\delta \geq 3$. Then*

$$\mathrm{pw}(G) \leq \sum_{v \in V(G)} g_{1/6}(d(v)) + \tau(n) + O(\log n).$$

Proof. We will show that $\mathrm{pw}(G) \leq f^-_{1/6}(G) + \tau(n) + O(\log n)$.

The proof follows the same general pattern as that for Theorems 4.1 and 4.7 above, but there are some differences of detail.

We first show that for any basic graph G, we have

$$\mathrm{pw}(G) \leq f^-_{1/6}(G) + \tau(n) + O(\log n).$$

For a connected cubic graph G, $\mathrm{pw}(G) \leq n/6 - 1/6 + \tau(n) + O(\log n) = f^-_{1/6}(G) + \tau(n) + O(\log n)$ by the result of Fomin and Høie [12] above (this is where we use the negative term, since for a cubic graph $f^-_{1/6}(G) = n/6 - 1/6$). If $r(G - v)$ is empty, then $G - v$ is

series-parallel and so has pathwidth $O(\log n)$, so $\mathrm{pw}(G) = O(\log n)$ as required. If G is basic with connected components $G_1, \ldots, G_k$, then

$$\begin{aligned}\mathrm{pw}(G) &= \max_i \mathrm{pw}(G_i) \\ &\leq \max_i \left(f^-_{1/6}(G_i) + \tau(|V(G_i)|) + O(\log |V(G_i)|) \right) \\ &\leq f^-_{1/6}(G) + \tau(n) + O(\log n)\end{aligned}$$

(Note that the fact that we have a maximum rather than a sum here is crucial.)

Now for any graph G with minimum degree at least 3, there is a set X_G such that

$$|X_G| + f^-_{1/6}(r(G - X_G)) \leq f^-_{1/6}(G)$$

and $r(G - X_G)$ is basic and non-empty. It follows from the results of [21] that $\mathrm{pw}(G) \leq |X_G| + \mathrm{pw}(G - X_G)$ and that $\mathrm{pw}(G - X_G) = \mathrm{pw}(r(G - X_G)) + O(\log n)$, so we have

$$\begin{aligned}\mathrm{pw}(G) &\leq |X_G| + \mathrm{pw}(G - X_G) \\ &= |X_G| + \mathrm{pw}(r(G - X_G)) + O(\log n) \\ &\leq |X_G| + f^-_{1/6}(r(G - X_G)) + \tau(n) + O(\log n) \quad \text{(from above)} \\ &\leq f^-_{1/6}(G) + \tau(n) + O(\log n),\end{aligned}$$

as required.

Corollary 5.2 *Let G be a graph, and $r(G)$ be the reduced graph of G. Then*

$$\mathrm{pw}(G) \leq \sum_{v \in V(r(G))} g_{1/6}(d_{r(G)}(v)) + \tau(|V(G)|) + O(\log |V(G)|) = \sum_{d \geq 3} g_{1/6}(d) n_d + o(|V(G)|).$$

□

For average degree, we obtain the following:

Theorem 5.3 *Let G be a graph of average degree $d \geq 2$, with n vertices and m edges. Then if G is connected, or $d \geq 5$,*

$$\mathrm{pw}(G) \leq (g_{1/6}(d) + o(1))n = (2g_{1/6}(d)/d + o(1))m = (1 - \tfrac{7/2}{d+1} + O(1/d^3))n.$$

□

It follows easily from the results of Golovnev and Kutzkov [17] that for each constant $0 < \varepsilon < 1$ there exists a constant $d_\varepsilon = O(\exp(1/\varepsilon))$ such that if $d_\varepsilon \leq d \leq \frac{n}{1+\varepsilon}$, then $\mathrm{pw}(G) \leq (1 - \frac{\varepsilon}{4}\frac{\ln d}{d})n$ for a graph G with n vertices and average degree d. For very large average degrees this gives a better bound.

The bound for pathwidth from Theorem 5.3 leads immediately to improved upper bounds for the time complexity of Scott and Sorkin's Algorithm C (for convenience of comparison we will call this Algorithm F, but it is really the same algorithm).

Theorem 5.4 *Let G be a graph, and $r(G)$ the reduced graph of G. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{\beta(G)}),$$

where $\beta(G) = \sum_{v \in V(r(G))} g_{1/6}(d_{r(G)}(v)) + \tau(|V(r(G)|)$. □

We also have

Theorem 5.5 *Let G be a connected graph with n vertices and m edges, of average degree $d \geq 2$. Then an instance of Max 2-CSP on graph G can be solved in time*

$$O^\star(r^{(g_{1/6}(d)+o(1))n}) = O^\star(r^{(2g_{1/6}(d)/d+o(1))m}) = O^\star(r^{(1-\frac{7/2}{d+1}+O(1/d^3))n}).$$

□

6 Fragmentability and treewidth

We have seen that one approach to obtaining faster algorithms for Max 2-CSP is to delete vertices from the graph in order to obtain a (preferably large) subgraph of bounded treewidth. We now show how the concept of fragmentability provides an upper bound on how much can be gained by this approach.

The concept of *fragmentability* of a class of graphs was introduced in [7] and surveyed in [9]. It measures how small a proportion of vertices needs to be removed from graphs in a class in order to break them into components of bounded size. We begin by recalling the definition.

Let $\varepsilon \in [0, 1]$ and $C \in \mathbb{N}$. A graph G is (C, ε)-*fragmentable* if there exists $X \subseteq V(G)$ such that $|X| \leq \varepsilon\,|V(G)|$ and every component of $G - X$ has at most C vertices. X is here called the *fragmenting set*. A class Γ of graphs is ε-*fragmentable* if there exists $C \in \mathbb{N}$ such that every $G \in \Gamma$ is (C, ε)-fragmentable. The *coefficient of fragmentability* of Γ is

$$c_f(\Gamma) = \inf\{\varepsilon \mid \Gamma \text{ is } \varepsilon\text{-fragmentable}\}.$$

Now consider the problem of removing vertices from a graph to make the treewidth bounded. We make the following (temporary) definition. Let Γ be a class of graphs. Define $c_{\mathrm{tw}}(\Gamma)$ by

$$c_{\mathrm{tw}}(\Gamma) = \inf\{\lambda : \exists t, \forall G \in \Gamma, \exists X \subseteq V(G), |X| \leq \lambda |V(G)| \text{ and } \mathrm{tw}(G - X) \leq t\}.$$

Thus, informally, $c_{\mathrm{tw}}(\Gamma)$ is the smallest proportion of the vertices (actually the infimum) whose removal from a graph in Γ is guaranteed to achieve (some fixed) bounded treewidth.

However it is easy to see that $c_{\mathrm{tw}}(\Gamma) = c_f(\Gamma)$. For suppose first that $\lambda > c_f(\Gamma)$. Then by definition of c_f, there is a constant C such that for any $G \in \Gamma$, there is a set $X \subseteq V(G)$ with $|X| \leq \lambda |V(G)|$, such that $G - X$ has all components of size at most C vertices, and

hence is of treewidth at most $C-1$. It follows immediately that $c_{\rm tw}(\Gamma) \leq \lambda$, and since λ was an arbitrary number greater than $c_f(\Gamma)$, we have $c_{\rm tw}(\Gamma) \leq c_f(\Gamma)$.

Conversely, suppose that $\lambda > c_{\rm tw}(\Gamma)$. Then for some fixed t, given any graph G in Γ, we can find a set $X \subseteq V(G)$, with $|X| \leq \lambda|(V(G)|$, such that $G-X$ has treewidth at most t. But graphs of treewidth t have small separators and so have coefficient of fragmentability 0, and so it follows [7] that $c_f(\Gamma) \leq \lambda$. Again, since λ was arbitrary, we have $c_f(\Gamma) \leq c_{\rm tw}(\Gamma)$.

Now let $\overline{\Gamma}^c_d$ be the class of connected graphs with average degree at most d. It was shown by Haxell *et al.* [19] (using connected regular claw-free examples) that for integer $d \geq 4$

$$c_f(\overline{\Gamma}^c_d) \geq \begin{cases} 1-\dfrac{4}{d+2}, & \text{if } d \text{ is even;} \\ 1-\dfrac{4(d+2)}{(d+1)(d+3)}, & \text{if } d \text{ is odd.} \end{cases}$$

Thus from above we know that in order to obtain a graph of bounded treewidth from a graph in $\overline{\Gamma}^c_d$, we have to delete (in the worst case) roughly a proportion $(1-\frac{4}{d+2})$ of the vertices. This compares with the proportion achieved by the algorithms in Section 4, which is roughly $(1-\frac{13/4}{d+1})n$ (or $(1-\frac{7/2}{d+1})n$ for claw-free graphs).

Thus an algorithm which deleted a smaller proportion of the vertices to obtain a graph of bounded treewidth would give a faster algorithm for Max 2-CSP. This technique will not be able to do better than replace $\frac{13}{4}$ by 4, nonetheless that would be a significant improvement given the difficulty of the problem.

7 Comparisons

We present here a brief comparison of our algorithms E and F with the earlier algorithms B,C of Scott and Sorkin, Algorithm D of Golovnev and Kutzkov and the exponential-space algorithm (W) of Williams [30], which solves constraint satisfaction problems in time $O^\star(r^{\omega n/3})$, where ω is the matrix multiplication constant ($\omega \approx 2.373$), and remains the fastest exponential space algorithm for larger degrees.

We compare each of the algorithms on graphs of average degree d, with n vertices, and give below the value of the exponent $\varepsilon(d)$ such that the complexity of the algorithm is $O^\star(r^{\varepsilon(d)n})$ (ignoring the $o(1)$ term in the exponent for algorithms B and F). In each case we show the fastest polynomial-space algorithm(s) in bold and the fastest exponential-space algorithm in italic.

	$\varepsilon(d)$ for algorithm					
d	B	C	D	E	F	W
3	**0.250**	*0.167*	**0.250**	**0.250**	*0.167*	0.791
4	**0.375**	*0.334*	0.400	**0.375**	*0.333*	0.791
5	**0.475**	*0.434*	0.500	**0.475**	*0.434*	0.791
6	0.570	0.520	0.572	**0.546**	*0.512*	0.791
7	0.665	0.607	0.625	**0.601**	*0.570*	0.791
8	0.760	0.694	0.667	**0.644**	*0.617*	0.791
9	0.855	0.780	0.700	**0.679**	*0.654*	0.791
10	0.950	0.867	0.728	**0.708**	*0.685*	0.791
11	1.045	0.954	0.750	**0.732**	*0.711*	0.791
12	1.140	1.040	0.770	**0.752**	*0.733*	0.791
13	1.235	1.127	0.786	**0.770**	*0.752*	0.791
14	1.330	1.214	0.800	**0.785**	*0.768*	0.791
15	1.425	1.300	0.813	**0.798**	*0.783*	0.791
16	1.520	1.387	0.824	**0.810**	0.795	*0.791*

Thus our Algorithm E is faster than all other polynomial-space algorithms for $d > 5$, and faster than all previous exponential-space algorithms when $6.810 \leq d \leq 14.496$. The exponential-space Algorithm F is slightly faster, but still beaten by Williams' exponential-space algorithm for $d \geq 15.694$.

Expressing the complexity in terms of the number of edges m, we give below, for graphs of average degree d, the value of the exponent $\eta(d)$ such that the complexity of the algorithm is $O^\star(r^{\eta(d)m})$:

	$\eta(d)$ for algorithm					
d	B	C	D	E	F	W
3	0.190	0.174	0.167	0.167	0.112	0.528
4	0.190	0.174	0.200	0.188	0.167	0.396
5	0.190	0.174	0.200	0.190	0.174	0.317
6	0.190	0.174	0.191	0.182	0.171	0.264
7	0.190	0.174	0.179	0.172	0.163	0.227
8	0.190	0.174	0.167	0.161	0.155	0.198
9	0.190	0.174	0.156	0.151	0.146	0.176
10	0.190	0.174	0.146	0.142	0.137	0.159
11	0.190	0.174	0.137	0.133	0.130	0.144
12	0.190	0.174	0.129	0.126	0.123	0.132
13	0.190	0.174	0.121	0.119	0.116	0.122
14	0.190	0.174	0.115	0.113	0.110	0.114
15	0.190	0.174	0.109	0.107	0.105	0.106
16	0.190	0.174	0.103	0.102	0.100	0.099

8 Appendix

In this section we prove the properties of the functions g_α and g'_α which are listed in Section 3.1. First recall the definition of the functions concerned:

For any $n \geq 2$, and α with $0 \leq \alpha \leq 1$, define the function $g_\alpha(n)$ by setting $g_\alpha(2) = 0$, $g_\alpha(3) = \alpha$, and for any $n \geq 4$,

$$ng_\alpha(n) = (n-2)g_\alpha(n-1) + g_\alpha(n-2) + 1. \tag{6}$$

We extend g_α to all real numbers at least 2 by linear interpolation, i.e., if $r = n+x$, where $n \geq 2$ is an integer, and $0 \leq x \leq 1$, then we set $g_\alpha(r) = (1-x)g_\alpha(n) + xg_\alpha(n+1)$.

Also, for any $n \geq 3$, define $g'_\alpha(n) = g_\alpha(n) - g_\alpha(n-1)$.

Lemma 8.1 *For any integer $n \geq 2$,*

$$g_\alpha(n) = (4-3\alpha)\frac{A(n)}{n!} + (2-3\alpha)\frac{(-1)^n}{n!} - (3-3\alpha)$$

where $A(n)$ is the alternating factorial function given by

$$A(n) = n! - (n-1)! + \ldots - (-1)^n \cdot 1!.$$

Proof. Let $h(n) = 1 - g_\alpha(n)$. Then from the recurrence (6) above we obtain, for $n \geq 4$,

$$n(1-h(n)) = (n-2)(1-h(n-1)) + 1 - h(n-2) + 1$$

or

$$nh(n) = (n-2)h(n-1) + h(n-2),$$

from which

$$nh(n) + h(n-1) = (n-1)h(n-1) + h(n-2)$$

Thus $nh(n)+h(n-1)$ is a constant, K say, for all $n \geq 3$. Multiplying through by $(n-1)!$, we obtain

$$n!h(n) + (n-1)!h(n-1) = K(n-1)!$$

or (replacing n by $n+1$)

$$(n+1)!h(n+1) + n!h(n) = Kn!$$

for $n \geq 2$. Setting $j(n) = (n+1)!h(n+1)/K$ for $n \geq 1$, we obtain

$$j(n) + j(n-1) = n!$$

for all $n \geq 2$. The alternating factorial function $A(n)$ satisfies

$$A(n) + A(n-1) = n!$$

so that

$$j(n) - A(n) = -(j(n-1) - A(n-1)).$$

Hence $j(n) - A(n) = C(-1)^n$ for some constant C. Thus

$$h(n) = (K/n!)j(n-1) = (K/n!)A(n-1) - (KC/n!)(-1)^n$$

or

$$g_\alpha(n) = 1 - (K/n!)(n! - A(n)) + (KC/n!)(-1)^n.$$

Thus

$$g_\alpha(n) = (K/n!)A(n) + (KC/n!)(-1)^n - (K-1).$$

Recall that $K = (n+1)h(n+1) + h(n)$, so setting $n = 2$ gives $K = 3(1-\alpha)+1 = 4-3\alpha$. Also $C = A(1) - j(1) = 1 - 2h(2)/K$, so $KC = K - 2h(2) = 4 - 3\alpha - 2 = 2 - 3\alpha$. So finally,

$$g_\alpha(n) = (4-3\alpha)\frac{A(n)}{n!} + (2-3\alpha)\frac{(-1)^n}{n!} - (3-3\alpha)$$

as required.

□

Lemma 8.2 *For all real $d \geq 2$,*

$$g_\alpha(d) = 1 - \frac{4-3\alpha}{d+1} + O(1/d^3).$$

Proof. We have, for $n \geq 2$,

$$\begin{aligned}
g_\alpha(n) &= (4-3\alpha)\frac{A(n)}{n!} + (2-3\alpha)\frac{(-1)^n}{n!} - (3-3\alpha) \\
&= (4-3\alpha)\frac{n! - (n-1)! + (n-2)! - A(n-3)}{n!} + (2-3\alpha)\frac{(-1)^n}{n!} - (3-3\alpha) \\
&= (4-3\alpha)\frac{n! - (n-1)! + (n-2)!}{n!} - (3-3\alpha) + O(1/n^3) \\
&= 1 - (4-3\alpha)\left(\frac{1}{n} - \frac{1}{n(n-1)}\right) + O(1/n^3) \\
&= 1 - \frac{(4-3\alpha)}{n+1} + (4-3\alpha)\left(\frac{1}{n+1} - \frac{1}{n} + \frac{1}{n(n-1)}\right) + O(1/n^3) \\
&= 1 - \frac{(4-3\alpha)}{n+1} + (4-3\alpha)\left(\frac{2}{n(n-1)(n+1)}\right) + O(1/n^3) \\
&= 1 - \frac{(4-3\alpha)}{n+1} + O(1/n^3).
\end{aligned}$$

The extension to non-integer arguments is straightforward. □

Lemma 8.3 *If $\alpha \leq 1/2$, then g_α is non-decreasing, i.e.,*

$$g_\alpha(n+1) \geq g_\alpha(n)$$

for all integers $n \geq 2$.

Proof. This is proved by induction, exactly as for the function $g = g_{1/4}$ in [8]. First a very easy induction shows that $g_\alpha < 1$ for all n. Then we show by induction that for all $n \geq 3$,

$$g_\alpha(n-1) \leq g_\alpha(n) \leq (1 + g_\alpha(n-1))/2$$

This is true for $n = 3$, so suppose it is true for some n. Then firstly,

$$(n+1)g_\alpha(n+1) = (n-1)g_\alpha(n) + g_\alpha(n-1) + 1 \geq (n-1)g_\alpha(n) + 2g_\alpha(n) = (n+1)g_\alpha(n),$$

and secondly

$$(n+1)g_\alpha(n+1) = (n-1)g_\alpha(n) + g_\alpha(n-1) + 1 \le ng_\alpha(n) + 1$$

so that

$$g_\alpha(n+1) \le \frac{n}{n+1}g_\alpha(n) + \frac{1}{n+1} \le \frac{1}{2}g_\alpha(n) + \frac{1}{2}$$

since $n \ge 1$ and $g_\alpha(n) < 1$. □

Note that if $\alpha > 1/2$, g_α is not non-decreasing, since $g_\alpha(4) = (1+2\alpha)/4 < \alpha = g_\alpha(3)$.

Lemma 8.4 *If $1/6 \le \alpha \le 3/10$, then g'_α is non-increasing, i.e.,*

$$g'_\alpha(n+1) \le g'_\alpha(n)$$

for all integers $n \ge 3$.

Proof. We have

$$\begin{aligned} ng_\alpha(n) &= (n-2)g_\alpha(n-1) + g_\alpha(n-2) + 1, \\ (n-1)g_\alpha(n-1) &= (n-3)g_\alpha(n-2) + g_\alpha(n-3) + 1 \end{aligned}$$

from which we obtain

$$ng'_\alpha(n) = (n-3)g'_\alpha(n-1) + g'_\alpha(n-2).$$

for $n \ge 5$. Then it is easily shown by an induction similar to the one above that

$$g'_\alpha(n-1) \ge g'_\alpha(n) \ge g'_\alpha(n-1)/3.$$

for all $n \ge 4$. The base case requires that $g'_\alpha(3) \ge g'_\alpha(4) \ge g'_\alpha(3)/3$, or $\alpha \ge (1-2\alpha)/4 \ge \alpha/3$, which is true since $1/6 \le \alpha \le 3/10$. □

Lemma 8.5 *If $1/6 \le \alpha \le 3/10$, $g_\alpha(n)/n$ is strictly decreasing for $n \ge 5$.*

Proof. First note that $g_\alpha(6) = (50\alpha + 53)/120 > 1/2$. Now $g_\alpha(n+1)/(n+1) < g_\alpha(n)/n$ if and only if $ng_\alpha(n+1) < (n+1)g_\alpha(n)$. Since

$$(n+1)g_\alpha(n+1) = (n-1)g_\alpha(n) + g_\alpha(n-1) + 1,$$

we have

$$ng_\alpha(n+1) = (n-1)g_\alpha(n) + g_\alpha(n-1) - g_\alpha(n+1) + 1,$$

so $ng_\alpha(n+1) < (n+1)g_\alpha(n)$ if and only if

$$g_\alpha(n-1) - g_\alpha(n+1) + 1 < 2g_\alpha(n),$$

or

$$2g_\alpha(n+1) - g'_\alpha(n+1) + g'_\alpha(n) > 1,$$

which is true since $g_\alpha(n+1) \ge g_\alpha(6) > 1/2$ and g'_α is non-increasing. In fact $g_\alpha(n)/n$ attains its maximum at $n = 5$ for all α in this range. □

Acknowledgements

We would like to thank David Wood for reminding us of the connection between planarization and Max 2-CSP, and the anonymous referees for their helpful comments and for drawing to our attention some recent work which we were unaware of.

References

[1] S. Arnborg, A. Proskurowski and D. G. Corneil, Forbidden minors characterization of partial 3-trees, *Discrete Math.* **80** (1990) 1–19.

[2] N. Bansal and V. Raman, Upper bounds for MAXSAT: further improved, in *Algorithms and computation* (Chennai, 1999), Lecture Notes in Computer Science **1741**, Springer, Berlin, pp. 247–258.

[3] H. L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, *Theoretical Computer Science* **209** (1998) 1–45.

[4] C. Cheng, E. McDermid and I. Suzuki, Planarization and Acyclic Colorings of Subcubic Claw-Free Graphs, in: P. Kolman and J. Kratochvíl (eds.), *Graph-Theoretic Concepts in Computer Science, WG 2011* (Czech Republic, June 2011), Lecture Notes in Computer Science **6986**, Springer, Berlin, 2011, pp. 107–118.

[5] H. de Fraysseix and P. Ossona de Mendez, A characterization of DFS cotree critical graphs, in: P. Mutzel, M. Jünger and S. Leipert (eds.), *Graph Drawing 2001* (Vienna, 23–26 Sept. 2001), Lecture Notes in Computer Science **2265**, Springer, Berlin, 2002, pp. 84–95.

[6] F. Della Croce, M. J. Kaminski and V. Th. Paschos, An exact algorithm for MAX-CUT in sparse graphs, *Oper. Res. Lett.* **35** (2007), 403–408.

[7] K. J. Edwards and G. E. Farr, Fragmentability of graphs, *J. Combin. Theory (Ser. B)* **82** (2001) 30–37.

[8] K. J. Edwards and G. E. Farr, Improved upper bounds for planarization and series-parallelization of average degree bounded graphs, *Electronic Journal of Combinatorics* **19 (2)** (2012) #P25.

[9] K. J. Edwards and G. E. Farr, Graph fragmentability, in: L. W. Beineke and R. J. Wilson (eds.), *Topics in Structural Graph Theory*, Encyclopedia of Mathematics and its Applications No. 147, Cambridge University Press, 2013, pp. 203–218. ISBN 978-0-521-80231-4.

[10] F. V. Fomin, S. Gaspers, S. Saurabh and A. A. Stepanov, On Two Techniques of Combining Branching and Treewidth, *Algorithmica* **54** (2009) 181–207.

[11] F. V. Fomin, F. Grandoni and D. Kratsch, A measure & conquer approach for the analysis of exact algorithms, *J. ACM* **56** (2009) 1–32.

[12] F. V. Fomin and K. Høie, Pathwidth of cubic graphs and exact algorithms, *Inf. Process. Lett.* **97** (2006) 191–196.

[13] F. V. Fomin and D. Kratsch, Exact Exponential Algorithms. Texts in Theoretical Computer Science. Springer, 2010. ISBN 978-3-642-16533-7

[14] S. Gaspers, Exponential Time Algorithms: Structures, Measures, and Bounds. VDM Verlag Dr. Mueller e.K., 2010. ISBN 978-3-639-21825-1
`http://www.cse.unsw.edu.au/~sergeg/SergeBookETA2010_screen.pdf`

[15] S. Gaspers and G. B. Sorkin, A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between, *J. Comput. System Sci.* **78** (2012), 305–335.

[16] A. Golovnev, New upper bounds for MAX-2-SAT and MAX-2-CSP w.r.t. the average variable degree, in: *Parameterized and exact computation*, Lecture Notes in Computer Science, 7112 Springer, Heidelberg, pp. 106–117.

[17] A. Golovnev and K. Kutzkov, New exact algorithms for the 2-constraint satisfaction problem, *Theoret. Comput. Sci.* **526** (2014), 18–27.

[18] J. Gramm, E. A. Hirsch, R. Niedermeier and P. Rossmanith, Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT, *Discrete Appl. Math.* **130** (2003) 139–155.

[19] P. Haxell, O. Pikhurko and A. Thomason, Maximum acyclic and fragmented sets in regular graphs, *J. Graph Theory* **57** (2008) 149–156.

[20] E. A. Hirsch, A new algorithm for MAX-2-SAT, in: *STACS 2000* (Lille), Lecture Notes in Computer Science **1770**, Springer, Berlin, 2000, pp. 65–73.

[21] J. Kneis, D. Mölle, S. Richter and P. Rossmanith, A bound on the pathwidth of sparse graphs with applications to exact algorithms, *SIAM Journal on Discrete Mathematics* **23** (2009) 407–427.

[22] A. Kojevnikov and A. S. Kulikov, A new approach to proving upper bounds for MAX-2-SAT, in: *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms 2006*, ACM, New York, 2006, pp. 11–17.

[23] A. S. Kulikov, K. Kutzkov, New bounds for MAX-SAT by clause learning, in: *Proceedings of the 2nd International Symposium on Computer Science in Russia* (CSR 2007), Lecture Notes in Computer Science **4649**, Springer, 2007, pp. 194–204.

[24] R. Niedermeier and P. Rossmanith, New upper bounds for maximum satisfiability, *J. Algorithms* **36** (2000), 63–88.

[25] D. Raible and H. Fernau, A new upper bound for Max-2-SAT: a graph-theoretic approach, in: *Mathematical foundations of computer science 2008*, Lecture Notes in Computer Science **5162**, Springer, Berlin, 2008, pp. 551–562.

[26] J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[27] A. D. Scott and G. B. Sorkin, Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances, in: *Proc. 7th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM 2003)*, Lecture Notes in Computer Science **2764**, Springer, 2003, pp. 382–395.

[28] A. D. Scott and G. B. Sorkin, Linear-programming design and analysis of fast algorithms for Max 2-CSP, *Discrete Optimization* **4** (2007) 260–287.

[29] A. D. Scott and G. B. Sorkin. Polynomial constraint satisfaction problems, graph bisection, and the Ising partition function, *ACM Trans. Algorithms* **5** (2009) Art. 45.

[30] R. Williams, A new algorithm for optimal constraint satisfaction and its implications, in: J. Díaz *et al.* (eds.), *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP)* (Turku, Finland, 2004), Lecture Notes in Computer Science **3142**, Springer, 2004, pp. 1227–1237.

[31] G. J. Woeginger, Exact algorithms for NP-hard problems: a survey, in: *Combinatorial optimization—Eureka, you shrink!*, Lecture Notes in Computer Science **2570**, Springer, 2003, pp 185–207.

[32] G. J. Woeginger. Open problems around exact algorithms, *Discrete Appl. Math.* **156** (2008) 397–405.

[33] Mingyu Xiao and Hiroshi Nagamochi, Exact Algorithms for Maximum Independent Set, in: L. Cai, S.-W. Cheng, and T.-W. Lam (eds.), *ISAAC2013*, Lecture Notes in Computer Science **8283**, Springer, 2013, pp. 328–338.